WHAT IS CLAIMED IS:

1. A processor for offloading processing in a storage environment, comprising:

a processor interface that interfaces said processor to a network processor configured to perform a storage function; and

semaphore circuitry, coupled to said processor interface, that receives a signal from said network processor, and that controls a semaphore related to said signal for locking and unlocking access to data.

2. The processor of claim 1, wherein said semaphore circuitry manages a queue for access to said semaphore.

3. The processor of claim 2, wherein said semaphore circuitry receives a second signal from said network processor and removes a request from said queue in response to said second signal when said network processor no longer desires said semaphore.

4. The processor of claim 2, wherein said semaphore circuitry refrains from sending to said network processor a second signal indicating said semaphore is unavailable, whereby said network processor continues to wait for said semaphore and said semaphore circuitry maintains ordered access to said queue.

5. The processor of claim 1, wherein said signal comprises one of a plurality of access requests for one of a plurality of semaphores, wherein said semaphore circuitry manages said plurality of access requests in a plurality of queues, and wherein each of said plurality of queues corresponds to a respective one of said plurality of semaphores.

6. The processor of claim 1, wherein said semaphore circuitry comprises:

a command queue that stores said signal received from said network processor.

7. The processor of claim 1, wherein said semaphore is a structure in a hash array, and wherein said semaphore circuitry comprises:

a hash key generator that performs a hashing function on said signal for accessing said hash array.

8. The processor of claim 1, wherein said semaphore circuitry comprises:

an update engine that receives a second signal from said network processor relating to a first process thread on said network processor, releases a lock on said semaphore related to said second signal, and sends a third signal to said network processor associating said semaphore with a second process thread on said network processor.

9. The processor of claim 1, wherein said semaphore circuitry comprises:

a semaphore queue manager that manages a queue of a plurality of semaphores.

10. The processor of claim 1, wherein said semaphore circuitry manages a queue for access to said semaphore, wherein said semaphore circuitry comprises:

a hash key generator that performs a hashing function on said signal for accessing a hash array, and that generates a hash key related to said signal, wherein said semaphore is a structure in said hash array, and wherein said signal relates to a first process thread on said network processor; and

an update engine, coupled to said hash key generator, that receives said hash key, that releases a lock on said semaphore related to said hash key, and that sends a second signal to said network processor associating said semaphore with a second process thread on said network processor, wherein said first process thread and said second process thread are associated with said semaphore in said queue.

11. A method of controlling a processor for offloading processing in a storage environment, comprising the steps of:

receiving a signal from a network processor configured to perform a storage function;

controlling a semaphore related to said signal for locking and unlocking access to data.

12. The method of claim 11, wherein said step of controlling said semaphore includes:

managing a queue for access to said semaphore.

13. The method of claim 12, further comprising:

receiving a second signal from said network processor; and

removing a request from said queue in response to said second signal when said network processor no longer desires said semaphore.

14. The method of claim 12, further comprising:

refraining from sending to said network processor a second signal indicating said semaphore is unavailable, whereby said network processor continues to wait for said semaphore and said processor maintains ordered access to said queue.

15. The method of claim 11, wherein said signal comprises one of a plurality of access requests for one of a plurality of semaphores, and wherein said step of controlling said semaphore includes:

managing said plurality of access requests in a plurality of queues, wherein each of said plurality of queues corresponds to a respective one of said plurality of semaphores.

16. The method of claim 11, wherein said step of controlling said semaphore includes:

receiving a second signal from said network processor relating to a first process thread on said network processor;

releasing a lock on said semaphore related to said second signal; and

sending a third signal to said network processor associating said semaphore with a second process thread on said network processor.

17. The method of claim 11, wherein said signal comprises one of a plurality of access requests for one of a plurality of semaphores, and wherein said step of controlling said semaphore includes:

managing said plurality of access requests in a plurality of queues, wherein each of said plurality of queues corresponds to a respective one of said plurality of semaphores;

receiving a second signal from said network processor relating to a first process thread on said network processor;

releasing a lock on said semaphore related to said second signal; and

sending a third signal to said network processor associating said semaphore with a second process thread on said network processor.

18. A processor for offloading processing in a storage environment, comprising:

a processor interface that interfaces said processor to a network processor configured to perform a storage function; and

ordering circuitry, coupled to said processor interface, that tracks an incoming order of incoming frames received by said network processor and controls an outgoing order of outgoing frames transmitted by said network processor.

19. The processor of claim 18, wherein said ordering circuitry controls said outgoing order by comparing a stored frame number with an incoming frame number of an incoming frame.

20. The processor of claim 18, wherein said ordering circuitry receives a first signal from said network processor, verifies an order of a frame associated with said first signal, and sends a second signal to said network processor, wherein said second signal indicates one of that said frame corresponds to an in-order frame and that said frame corresponds to an out-of-order frame.

21. The processor of claim 20, wherein when said frame corresponds to said in-order frame, said ordering circuitry receives a third signal from said network processor after said network processor has transmitted said frame, and said ordering circuitry tracks said outgoing order in response to said third signal.

22. The processor of claim 20, wherein when said frame corresponds to said out-of-order frame, said ordering circuitry generates said second signal indicating that said frame is to be stored by said network processor.

23. The processor of claim 18, wherein said ordering circuitry receives a first signal from said network processor, determines whether a particular frame stored by the network

processor is in order to be transmitted, and identifies said particular frame to said network processor.

24. The processor of claim 18, wherein said ordering circuitry comprises:

a command processor that processes a command from said network processor relating to an incoming frame.

25. The processor of claim 18, wherein said ordering circuitry comprises:

an update state machine that manages a plurality of queues corresponding to a plurality of process threads of said network processor, wherein a queue of said plurality of queues includes a frame structure corresponding to a frame received by said network processor.

26. The processor of claim 18, wherein said ordering circuitry comprises:

a read buffer that stores data relating to a next frame for said network processor to transmit.

27. The processor of claim 18, wherein said ordering circuitry comprises:

a command processor that processes a command from said network processor relating to an incoming frame;

an update state machine, coupled to said command processor, that manages a plurality of queues corresponding to a plurality of process threads of said network processor, wherein a queue of said plurality of queues includes a frame structure corresponding to a frame received by said network processor; and

a read buffer, coupled to said update state machine, that stores data relating to a next frame for said network processor to transmit, as identified by said update state machine.

28. A method of controlling a processor for offloading processing in a storage environment, comprising the steps of:

tracking an incoming order of incoming frames received by said network processor; and

controlling an outgoing order of outgoing frames transmitted by said network processor.

29. The method of claim 28, wherein said step of controlling includes:

comparing a stored frame number with an incoming frame number of an incoming frame.

30. The method of claim 28, wherein said step of controlling includes:

receiving a first signal from said network processor;

verifying an order of a frame associated with said first signal; and

sending a second signal to said network processor, wherein said second signal indicates one of that said frame corresponds to an in-order frame and that said frame corresponds to an out-of-order frame.

31. The method of claim 30, wherein when said frame corresponds to said in-order frame, said step of controlling further includes:

receiving a third signal from said network processor after said network processor has transmitted said frame; and

tracking said outgoing order in response to said third signal.

32. The method of claim 30, wherein when said frame corresponds to said out-of-order frame, said step of controlling further includes:

generating said second signal indicating that said frame is to be stored by said network processor.

33. The method of claim 28, wherein said step of controlling includes:

receiving a first signal from said network processor;

determining whether a particular frame stored by the network processor is in order to be transmitted; and

identifying said particular frame to said network processor.

34. The method of claim 28, wherein said step of controlling includes:

processing a command from said network processor relating to an incoming frame;

managing a plurality of queues corresponding to a plurality of process threads of said network processor, wherein a queue of said plurality of queues includes a frame structure corresponding to a frame received by said network processor; and

storing data relating to a next frame for said network processor to transmit, as identified by said step of managing.

35. A processor for offloading processing in a storage environment, comprising:

a processor interface that interfaces said processor to a network processor configured to perform a storage function; and

timer circuitry, coupled to said processor interface, that receives a plurality of signals, that manages a plurality of timers related to said plurality of signals, and that generates a timing result when one of said plurality of timers is completed.

36. The processor of claim 35, wherein each of said plurality of timers corresponds to a respective element in a doubly-linked list of elements.

37. The processor of claim 36, wherein said timer circuitry manages said doubly-linked list by comparing a free-running timer value to a first value associated with a current element of said doubly-linked list, changing a second value associated with said current element when said first value matches said free-running timer value, associating said current element with an end of said doubly linked list after said second value has been changed, and moving on to a next element of said doubly-linked list.

38. The processor of claim 36, wherein said timer circuitry comprises:

a cache configured to store at least a portion of said doubly-linked list, wherein said portion includes a current element and a previous element.

39. The processor of claim 35, wherein said plurality of timers correspond to a plurality of respective elements in a plurality of doubly-linked lists of elements, and wherein each of said plurality of doubly-linked lists corresponds to a respective time base period.

40. The processor of claim 35, wherein said timer circuitry comprises:

an expired timer queue that stores a plurality of timing results corresponding to a plurality of expired timers.

41. The processor of claim 40, wherein said network processor periodically accesses said expired timer queue via said processor interface.

42. The processor of claim 40, wherein said timer circuitry sends an interrupt to trigger said network processor to access said expired timer queue.

43. The processor of claim 35, wherein said timer circuitry comprises:

an expired timer queue manager that controls filling said expired timer queue with said plurality of timing results from a memory.

44. The processor of claim 35, wherein said timer circuitry comprises:

a remover circuit that removes a timer from said plurality of timers in response to a stop timer signal via said processor interface from said network processor.

45. The processor of claim 35, wherein a first portion of said plurality of timers are stored in an external memory, and wherein said timer circuitry comprises:

a cache configured to store a second portion of said plurality of timers; and

an arbiter circuit that controls memory access between said external memory and said cache.

46. The processor of claim 35, wherein said plurality of timers correspond to a plurality of respective elements in a plurality of doubly-linked lists of elements, and wherein each of said plurality of doubly-linked lists corresponds to a respective time base period, wherein said processor further comprises:

an expired timer queue that stores a plurality of timing results corresponding to a plurality of expired timers; and

an expired timer queue manager that controls filling said expired timer queue with said plurality of timing results from a memory.

47. A method of controlling a processor for offloading processing in a storage environment, comprising the steps of:

receiving a signal from a network processor configured to perform a storage function;

managing a timer related to said signal; and

generating a timing result when said timer is completed.

48. The method of claim 47, wherein said step of managing said timer comprises:

adding said timer to a doubly-linked list, wherein said doubly-linked list comprises a plurality of timers.

49. The method of claim 47, wherein said step of managing said timer comprises:

adding said timer to one of a plurality of doubly-linked lists, wherein each of said plurality of doubly-linked lists comprises a plurality of timers, and wherein each of said plurality of doubly-linked lists corresponds to a respective time base period.

50. The method of claim 47, further comprising:

filling an expired timing queue with said timing result.

51. The method of claim 47, further comprising:

stopping said timer in response to a stop timer signal from said network processor.